Buku Ajar & Modul
Praktikum Perkuliahan

SISTEM INFORMASI

ITATS
INSTITUT TEKNOLOGI ADHI TAMA SURABAYA

REKAYASA PERANGKAT LUNAK

"Prinsip Layered Technology, Model SDLC (Waterfall, V-Model, Spiral, Agile Scrum), Rekayasa Kebutuhan (SRS IEEE 830), Pemodelan UML, Pengujian (Black/White Box), & Manajemen Proyek Berbasis RPS"

Proses Terstruktur

Pemodelan Sistem

Kualitas & Manajemen Proyek

Sulistyowati, S.T., M.Kom.
Rachman Arief, S.Kom., M.Kom.

Edisi Pertama, Agustus 2024

BUKU AJAR & MODUL PERKULIAHAN

REKAYASA PERANGKAT LUNAK (KODE MK: 25135024)

Buku Ajar & Modul Perkuliahan Teori, Prinsip Layered Technology, Model SDLC (Waterfall, V-Model, Spiral, Agile Scrum), Rekayasa Kebutuhan (SRS IEEE 830), Pemodelan UML, Pengujian (Black/White Box), & Manajemen Proyek Berbasis RPS

Dosen Pengembang RPS & Pengampu:

Sulistyowati, S.T., M. Kom. & Rachman Arief, S. Kom., M. Kom.

Bobot: 3 SKS (Semester 5 - Teori)
Program Studi S1 Sistem Informasi
Fakultas Teknik Elektro dan Teknologi Informasi
Institut Teknologi Adhi Tama Surabaya (ITATS)
Edisi Pertama, Tahun Akademik 2024/2025

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya sehingga buku ajar dan modul perkuliahan "Rekayasa Perangkat Lunak (Kode MK: 25135024)" ini dapat diselesaikan dengan baik.

Perkembangan teknologi komputasi dan transformasi digital enterprise menempatkan perangkat lunak sebagai komponen paling kritis dalam menggerakkan proses bisnis modern. Namun, membangun perangkat lunak berkualitas tinggi, andal, mudah dipelihara (maintainable), dan selesai tepat waktu bukanlah sekadar aktivitas pengkodean (coding) semata. Rekayasa Perangkat Lunak (Software Engineering) adalah disiplin teknik yang menerapkan pendekatan sistematis, terstruktur, disiplin, dan terukur dalam proses pengembangan, pengoperasian, dan pemeliharaan perangkat lunak (Pressman & Maxim, 2020; Sommerville, 2016).

Buku modul perkuliahan ini disusun secara komprehensif mengacu pada Rencana Pembelajaran Semester (RPS) resmi Program Studi S1 Sistem Informasi ITATS, mencakup 5 Bahan Kajian Pokok yang dipetakan ke dalam 5 Bab Utama:

1. Konsep Dasar & Prinsip-Prinsip Rekayasa Perangkat Lunak (Layered Technology, Krisis Software, Dual Role of Software).
2. Model Proses Pengembangan Perangkat Lunak (SDLC: Waterfall Preskriptif, V-Model, Prototyping, Spiral, & Agile Scrum Framework).
3. Rekayasa Kebutuhan (Requirements Engineering: Elisitasi Kebutuhan, Analisis FURPS+, & Dokumentasi SRS IEEE 830 / ISO 29148).

4. Perancangan Perangkat Lunak & Pemodelan Sistem Berbasis UML (Use Case, Activity, Class, Sequence Diagram, & Arsitektur MVC / Microservices).
5. Manajemen Proyek Perangkat Lunak, Pengujian (Black-Box & White-Box Testing), serta Penyusunan Dokumen SRS Proyek Akhir.

Buku ini juga menyajikan panduan penugasan terstruktur resmi RPS (Tugas 1 Literature Review Paper 3%, UTS 7%, Tugas 2 Model SDLC 15%, Tugas 3 Elisitasi & Pemodelan UML 20%, Tugas 4 Manajemen Proyek SRS 15%, dan UAS Presentasi Proyek 40% dengan total bobot kumulatif 100%).

Penulis menyampaikan terima kasih yang setinggi-tingginya kepada Ketua Program Studi Sistem Informasi ITATS, pimpinan Fakultas Teknik Elektro dan Teknologi Informasi, serta rekan-rekan dosen atas dukungan dan diskusinya. Semoga modul perkuliahan ini bermanfaat dalam mencetak Sarjana Sistem Informasi yang kompeten sebagai Analis Sistem (System Analyst), Software Architect, maupun Manajer Proyek TI yang handal.

Kritik dan saran konstruktif senantiasa kami nantikan demi penyempurnaan di masa mendatang.

Surabaya, Agustus 2024

Tim Dosen Pengampu,
Sulistyowati, S.T., M.Kom. & Rachman Arief, S.Kom., M.Kom.

Capaian Pembelajaran Lulusan (CPL) & CPMK

Kode	Deskripsi Capaian Pembelajaran (RPS Resmi)
CPL-7	Mampu memahami dan menggunakan berbagai metodologi pengembangan sistem beserta alat permodelan serta menganalisis kebutuhan pengguna dalam membangun sistem informasi yang berkualitas untuk mencapai tujuan organisasi dan/atau enterprise.
CPMK-1 (100%)	Mampu memahami berbagai metodologi pengembangan sistem beserta prinsip rekayasa kebutuhan, pemodelan UML, dan manajemen proyek perangkat lunak.

Pemetaan Sub-CPMK & Bobot Penilaian Tugas RPS

Sub-CPMK & Bobot	Kemampuan Akhir Tiap Tahapan Belajar
SubCPMK 1 (Minggu 1-2 | 5%)	Mahasiswa mampu menjelaskan konsep dasar dan prinsip-prinsip rekayasa perangkat lunak dalam konteks pengembangan sistem informasi [C2, A2, P2].
SubCPMK 2 (Minggu 3-8 | 20%)	Mahasiswa mampu memahami proses dan tahapan dalam berbagai model pengembangan perangkat lunak (SDLC) [C2, A2, P2].
SubCPMK 3 (Minggu 9-10 | 20%)	Mahasiswa mampu menganalisis kebutuhan pengguna dengan teknik elicitation dan menyusun dokumen rekayasa kebutuhan yang lengkap dan sistematis [C4, A4, P4].
SubCPMK 4 (Minggu 11-13 | 20%)	Mahasiswa mampu membuat model sistem menggunakan diagram UML (Unified Modeling Language) sebagai bagian dari dokumentasi perangkat lunak [C3, A3, P3].
SubCPMK 5 (Minggu 14-16 | 35%)	Mahasiswa mampu menganalisis kebutuhan proyek perangkat lunak, memilih model SDLC yang sesuai dengan konteks proyek, serta menyusun dokumen SRS sebagai dasar perencanaan proyek [C4, A3, P4].

Roadmap Perkuliahan 16 Pertemuan & Korelasi Penugasan RPS

Minggu	Bahan Kajian & Topik Perkuliahan	Bentuk Evaluasi / Tugas
Minggu 1-2	Konsep Dasar & Prinsip Rekayasa Perangkat Lunak (Bahan Kajian 1 / SubCPMK 1)	Kuliah Teori, Diskusi Krisis Software, Tugas 1: Literature Review Paper (3%)

Minggu	Bahan Kajian & Topik Perkuliahan	Bentuk Evaluasi / Tugas
Minggu 3-5	Model SDLC Preskriptif: Waterfall, V-Model, Prototyping, Spiral (Bahan Kajian 2 / SubCPMK 2)	Kuliah Interaktif, Diskusi Karakteristik SDLC Linier vs Risk-Driven
Minggu 6-7	Metodologi Agile & Scrum Framework (Bahan Kajian 2 / SubCPMK 2)	Studi Kasus Sprints & User Stories, Tugas 2: Presentasi Model SDLC (15%)
Minggu 8	Evaluasi Tengah Semester (UTS)	Ujian Teori Tertulis Konsep RPL & Model SDLC (Bobot 7% kumulatif)
Minggu 9-10	Rekayasa Kebutuhan & Spesifikasi SRS (Bahan Kajian 3 / SubCPMK 3)	Teknik Elisitasi Kebutuhan, Analisis FURPS+, Format Standar SRS IEEE 830
Minggu 11-13	Perancangan Sistem & Pemodelan UML (Bahan Kajian 4 / SubCPMK 4)	Use Case, Activity, Class, Sequence Diagram, Tugas 3: Dokumen Elisitasi & UML (20%)
Minggu 14-15	Manajemen Proyek, Estimasi, & Pengujian Software (Bahan Kajian 5 / SubCPMK 5)	WBS, Function Points, Black/White Box Testing, Tugas 4: Manajemen Proyek SRS (15%)
Minggu 16	Evaluasi Akhir Semester (UAS)	Presentasi Final Proyek & Validasi Dokumen SRS Terpadu (Bobot 40% kumulatif)

DAFTAR ISI

BAB 1: KONSEP DASAR & PRINSIP REKAYASA PERANGKAT LUNAK

1.1 Hakikat dan Definisi Rekayasa Perangkat Lunak

Perangkat lunak (Software) bukan sekadar baris kode program komputer (Source Code), melainkan sebuah kesatuan terpadu yang mencakup: (1) Instruksi program yang mengeksekusi fungsionalitas dan kinerja yang diinginkan, (2) Struktur data yang memungkinkan program memanipulasi informasi secara memadai, dan (3) Dokumen deskriptif yang menjelaskan operasi dan penggunaan program (Pressman & Maxim, 2020).

Standar IEEE 610.12 mendefinisikan Rekayasa Perangkat Lunak (Software Engineering) sebagai: '(1) Penerapan pendekatan yang sistematis, disiplin, dan terukur terhadap pengembangan, pengoperasian, dan pemeliharaan perangkat lunak; yaitu penerapan prinsip-prinsip rekayasa (engineering) pada perangkat lunak, dan (2) Studi ilmiah mengenai pendekatan-pendekatan tersebut'.

Ian Sommerville (2016) menegaskan bahwa rekayasa perangkat lunak berfokus pada semua aspek produksi perangkat lunak mulai dari tahapan awal konseptualisasi sistem hingga fase pemeliharaan operasional pasca-implementasi.

1.2 Karakteristik Unik Perangkat Lunak

Perangkat lunak memiliki sifat-sifat fundamental yang sangat berbeda dari produk rekayasa fisik/manufaktur:

1. Software is Engineered or Developed, Not Manufactured: Perangkat lunak tidak diproduksi di pabrik perakitan massal

dengan bahan baku fisik. Biaya terbesar perangkat lunak terletak pada fase rekayasa perancangan logis (Design Phase), bukan pada proses replikasi salinan berkasnya.

2. Software Does Not 'Wear Out' (Tidak Aus Secara Fisik): Perangkat keras fisik mengalami keausan mekanis akibat gesekan, suhu, dan usia (mengikuti Kurva Bak Mandi / Bathtub Curve). Sebaliknya, perangkat lunak tidak pernah aus secara fisik; penurunan mutu software (Software Deterioration) terjadi akibat akumulasi perubahan kode yang tidak terencana (Side Effects) saat perbaikan bug atau perubahan kebutuhan lingkungan.
3. Custom-Built vs. Component-Based: Sebagian besar perangkat lunak enterprise masih dibangun secara khusus (Custom-Built), meskipun industri modern kini bertransisi cepat menuju perakitan komponen siap pakai (Component-Based Software Engineering / CBSE).

1.3 Krisis Perangkat Lunak & Urgensi Pendekatan Rekayasa

Pada era 1960-an hingga 1980-an, industri komputer menghadapi fenomena kegagalan masif yang dikenal sebagai Krisis Perangkat Lunak (The Software Crisis):

- Biaya pengembangan perangkat lunak membengkak berkali-kali lipat melebihi anggaran awal.
- Waktu penyelesaian molor bertahun-tahun melewati tenggat jadwal.
- Perangkat lunak yang dihasilkan penuh dengan cacat (Bugs) dan sulit dioperasikan oleh pengguna.

- Kode program sangat rumit (Spaghetti Code) sehingga mustahil untuk dipelihara atau dikembangkan lebih lanjut.

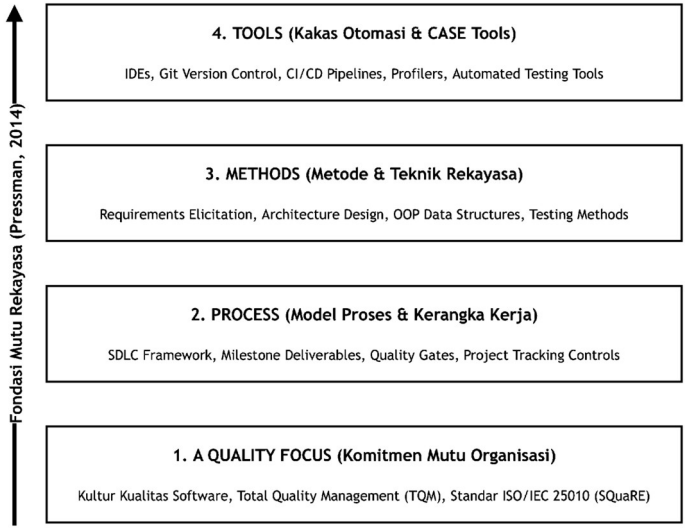
Krisis ini menyadarkan komunitas teknologi informasi global bahwa menulis program secara serampangan (Hacking / Cowboy Coding) tidak lagi memadai untuk sistem enterprise yang kompleks. Dibutuhkan metodologi rekayasa terstruktur dan terukur.

1.4 Arsitektur Rekayasa Perangkat Lunak Berlapis (Layered Technology)

Roger S. Pressman (2014) merumuskan bahwa rekayasa perangkat lunak bertumpu pada arsitektur berlapis (Layered Technology):

1. A Quality Focus (Fokus Kualitas): Lapisan fondasi paling dasar. Komitmen organisasi terhadap budaya mutu, standar rekayasa, dan manajemen kualitas total (TQM).
2. Process (Proses): Perekat yang menyatukan seluruh lapisan teknologi. Mendefinisikan kerangka kerja aktivitas (Framework Activities), tonggak capaian (Milestones), produk kerja (Deliverables), dan kontrol kualitas (Quality Gates) sepanjang siklus hidup pengembangan sistem.
3. Methods (Metode): Menyediakan teknik teknis 'How-To' dalam membangun perangkat lunak (Rekayasa Kebutuhan, Perancangan Arsitektur, Pemodelan UML, Pengkodean, dan Pengujian Sistem).
4. Tools (Kakas Otomasi): Perangkat bantu otomatis dan semi-otomatis (Computer-Aided Software Engineering / CASE Tools)

yang mendukung proses dan metode (seperti Git Version Control, Visual Studio Code, Jenkins CI/CD, dan Jira Project Tracking).



Gambar 1.1: Arsitektur Rekayasa Perangkat Lunak Berlapis (Layered Technology - Pressman)

Aktivitas Pembelajaran & Penugasan Terstruktur: LEMBAR KERJA 1 (TUGAS 1 RPS): Literature Review Paper Software Engineering

Kerjakan penugasan terstruktur mandiri 'Tugas 1 - Literature Review Paper' (Bobot 3%) dengan menelaah satu artikel jurnal ilmiah bereputasi (SINTA / Scopus / IEEE Xplore) mengenai topik rekayasa perangkat lunak kontemporer.

Bagian A: Identifikasi Metadata Paper Ilmiah yang Ditelaah

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
1. Judul Artikel Jurnal Ilmiah:	Analisis Perbandingan Metodologi Agile Scrum dan Waterfall pada Proyek Enterprise
2. Penulis & Nama Jurnal / Penerbit:	Pratama et al. (2024), Journal of Software Engineering & Information Systems
3. Latar Belakang Masalah (Research Gap):	Tingginya tingkat kegagalan rilis fitur software perbankan akibat perubahan regulasi cepat
4. Metodologi Penelitian yang Digunakan:	Studi Kasus Komparatif Empiris pada 10 Proyek Pengembangan Sistem Informasi
5. Kesimpulan Utama Paper:	Agile Scrum mereduksi Time-to-Market sebesar 45% dibanding Waterfall pada sistem dinamis

Bagian B: Evaluasi Penerapan Layered Technology pada Studi Kasus

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
Analisis 'A Quality Focus':	Penerapan standar ISO/IEC 25010 untuk mengukur reliabilitas modul transaksi
Analisis 'Process':	Penerapan framework Scrum dengan siklus Sprint 2 mingguan dan Daily Standup
Analisis 'Methods':	Elisitasi kebutuhan berbasis User Stories dan pemodelan Class/Sequence Diagram

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
Analisis 'Tools' yang Dipakai:	GitLab CI/CD untuk automated testing dan Docker untuk deployment container
Status Kelengkapan Berkas Tugas 1:	Dokumen Resume Literature Review Paper lengkap terlampir untuk penilaian SubCPMK 1

Lembar Refleksi & Evaluasi Proyek:

1. Jelaskan mengapa perangkat lunak tidak mengalami keausan fisik (Wear Out) seperti perangkat keras, namun dapat mengalami degradasi mutu (Deterioration) seiring berjalannya waktu:

.....

2. Bagaimana pemahaman terhadap 4 lapisan Layered Technology (Pressman) membedakan pola pikir seorang Software Engineer profesional dengan seorang Programmer pemula?

BAB 2: MODEL PROSES PENGEMBANGAN PERANGKAT LUNAK (SDLC)

2.1 Konsep Siklus Hidup SDLC & Generic Process Framework

Software Development Life Cycle (SDLC) adalah kerangka kerja terstruktur yang menggambarkan tahapan-tahapan yang dilalui perangkat lunak mulai dari konsepsi awal, perancangan, implementasi kode, pengujian, rilis, hingga fase pemeliharaan operasional.

Roger S. Pressman (2014) merumuskan 5 Aktivitas Kerangka Kerja Generik (Generic Framework Activities) yang ada pada setiap model SDLC:

1. Komunikasi (Communication): Berdialog intensif dengan pemangku kepentingan (klien dan calon pengguna) untuk menggali inisiasi proyek dan elisitasi kebutuhan sistem.
2. Perencanaan (Planning): Menetapkan estimasi biaya, sumber daya manusia, jadwal waktu pengerjaan (Gantt Chart), dan manajemen risiko proyek.
3. Pemodelan (Modeling): Merancang representasi visual arsitektur sistem, struktur basis data, antarmuka pengguna (UI/UX), dan diagram alur logika bisnis (UML).
4. Konstruksi (Construction): Aktivitas penulisan kode program (Coding) dan pengujian verifikasi modul (Unit Testing & Integration Testing).
5. Penerapan (Deployment): Pengiriman rilis perangkat lunak ke lingkungan produksi, evaluasi pengguna (User Acceptance Testing / UAT), dan penyediaan layanan dukungan teknis.

2.2 Model-Model Proses Preskriptif: Waterfall, V-Model, & Spiral

Model preskriptif menekankan pada keteraturan, struktur formal, dan dokumentasi yang ketat:

A. Waterfall Model (Model Air Terjun - Winston Royce, 1970):

Model linier sekuensial klasik di mana setiap fase harus diselesaikan 100% dan disetujui (Signed-Off) sebelum fase berikutnya boleh dimulai. Sangat cocok untuk proyek dengan kebutuhan yang sudah sangat jelas, stabil, dan tidak berubah-ubah (misal: perangkat lunak instrumen medis atau sistem penerbangan), namun tidak fleksibel terhadap perubahan di tengah jalan.

B. V-Model (Verification and Validation Model):

Variasi dari Waterfall yang menghubungkan setiap tahap perancangan di sisi kiri (Requirements, Architecture, Component Design) secara simetris dengan tahap pengujian di sisi kanan (Acceptance Testing, System Testing, Unit Testing).

C. Prototyping Model:

Membangun versi miniatur sistem yang cepat (Mockup / Prototipe Fungsional) untuk memperoleh umpan balik awal dari pengguna saat kebutuhan awal masih ambigu.

D. Spiral Model (Barry Boehm, 1988):

Model proses evolusioner yang menggabungkan sifat iteratif prototyping dengan kontrol sistematis Waterfall. Karakteristik pembeda utamanya adalah adanya analisis risiko formal (Risk Analysis) yang mendalam pada setiap putaran spiral sebelum keputusan investasi lanjutan dibuat.

2.3 Paradigma Metodologi Agile & Kerangka Kerja Scrum

Agile Software Development lahir dari Deklarasi Agile Manifesto (2001) yang menekankan pada adaptabilitas, kolaborasi intensif, dan pengiriman rilis perangkat lunak yang cepat dan bernilai secara bertahap.

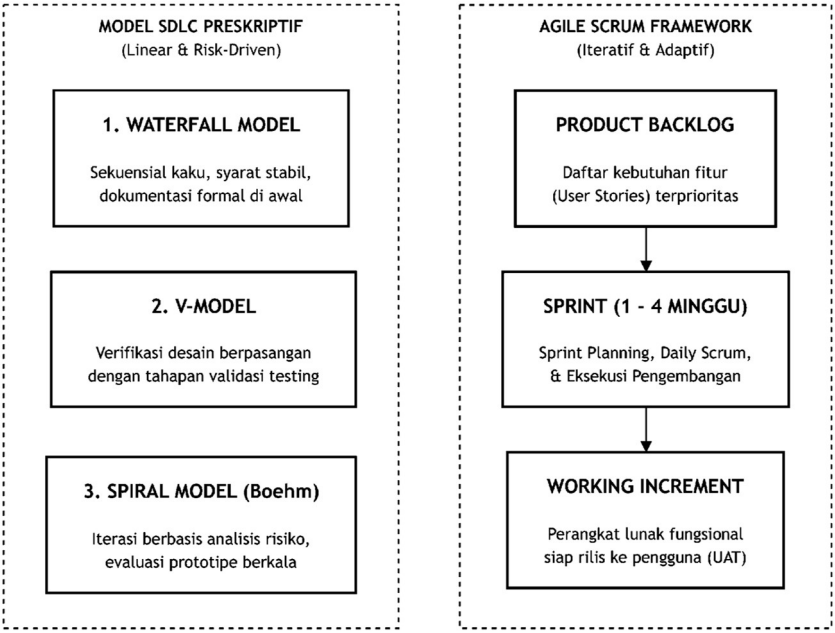
Empat Nilai Fundamental Agile Manifesto:

1. Individu dan interaksi LEBIH DARI proses dan kakas (tools).
2. Perangkat lunak yang berfungsi LEBIH DARI dokumentasi yang komprehensif.
3. Kolaborasi dengan pelanggan LEBIH DARI negosiasi kontrak.
4. Tanggap terhadap perubahan LEBIH DARI mengikuti rencana yang kaku.

Kerangka Kerja Agile Scrum (Ken Schwaber & Jeff Sutherland):

- Tiga Peran Scrum (Scrum Roles): (1) Product Owner (Pemilik visi produk & penentu prioritas fitur), (2) Scrum Master (Fasilitator proses & pelindung tim dari hambatan), (3) Developers / Tim Pengembang (Tim lintas fungsi yang mengeksekusi pembuatan software).
- Tiga Artefak Scrum (Scrum Artifacts): (1) Product Backlog (Daftar fitur User Stories yang terurut prioritas), (2) Sprint Backlog (Item pekerjaan yang dipilih untuk diselesaikan dalam satu Sprint), (3) Product Increment (Hasil perangkat lunak fungsional yang memenuhi Definition of Done).
- Empat Acara Scrum (Scrum Events): (1) Sprint Planning (Perencanaan Sprint), (2) Daily Scrum (Rapat koordinasi harian 15 menit), (3) Sprint Review (Demo hasil software ke stakeholder),

(4) Sprint Retrospective (Evaluasi proses tim untuk perbaikan berkelanjutan).



Gambar 2.1: Spektrum Model Proses SDLC (Waterfall, V-Model, Spiral, & Agile Scrum)

2.4 Matriks Pemilihan Model SDLC Sesuai Karakteristik Proyek

Pemilihan model SDLC harus didasarkan pada karakteristik proyek:

- Kebutuhan Jelas & Regulasi Ketat -> Waterfall atau V-Model.
- Kebutuhan Antarmuka Belum Jelas -> Prototyping Model.
- Proyek Berskala Raksasa dengan Risiko Teknis Tinggi -> Spiral Model.

- Lingkungan Bisnis Dinamis, Startup, & Time-to-Market Cepat -> Agile Scrum.

Aktivitas Pembelajaran & Penugasan Terstruktur: LEMBAR KERJA 2 (TUGAS 2 RPS & UTS): Analisis Komparasi Model SDLC & Sprint Scrum

Kerjakan penugasan terstruktur kelompok 'Tugas 2 - Model SDLC' (Bobot 15%) dan kumpulkan bahan ujian untuk Evaluasi Tengah Semester (UTS - Bobot 7% kumulatif).

Bagian A: Matriks Pemilihan Model SDLC untuk Kasus Nyata Enterprise

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
1. Judul Studi Kasus Proyek Perangkat Lunak:	Sistem Informasi Rekrutmen & Penilaian Kinerja Karyawan (HRIS Enterprise)
2. Karakteristik Kebutuhan Sistem:	Kebutuhan modul rekrutmen sudah jelas, modul KPI dinamis mengikuti regulasi direksi
3. Model SDLC yang Dipilih & Alasannya:	Agile Scrum (Memungkinkan rilis modul bertahap per Sprint 2 mingguan)
4. Analisis Risiko Kegagalan Utama:	Perubahan format indikator kinerja di tengah proyek (Dimediasi via Sprint Backlog)
5. Frekuensi Keterlibatan Stakeholder:	Tinggi (Product Owner dan Manajer HR hadir pada setiap Sprint Review demo)

Bagian B: Simulasi Rencana Sprint Scrum (Sprint Planning Skenario 1)

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
Durasi Waktu Sprint 1:	2 Minggu (10 Hari Kerja)

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
Daftar Item User Story di Sprint Backlog:	US-01: Login Multi-Role (5 Story Points) | US-02: Form Lowongan Kerja (8 Points)
Alokasi Peran Tim Scrum:	Product Owner: HR Lead | Scrum Master: Analis Sistem | Dev: 3 Fullstack Engineer
Kriteria Definition of Done (DoD):	Lulus Unit Test 100%, Code Review disetujui, dan lolos UAT di staging server
Status Kesiapan Berkas Tugas 2 & UTS:	Lengkap terlampir (Slide presentasi model SDLC & berkas evaluasi UTS)

Lembar Refleksi & Evaluasi Proyek:

1. Jelaskan dalam situasi proyek nyata seperti apa model Waterfall masih menjadi pilihan yang jauh lebih unggul dibandingkan model Agile Scrum:

.....

2. Bagaimana peran seorang Scrum Master dalam melindungi tim pengembang dari gangguan penambahan fitur mendadak di tengah jalannya Sprint?

BAB 3: REKAYASA KEBUTUHAN (REQUIREMENTS ENGINEERING) & SPESIFIKASI SRS

3.1 Hakikat dan Urgensi Rekayasa Kebutuhan (Requirements Engineering)

Kesalahan paling fatal dan paling mahal dalam rekayasa perangkat lunak bukanlah bug pada baris kode program, melainkan kegagalan dalam memahami apa yang sebenarnya dibutuhkan oleh pengguna (Karl Wiegers & Candase Hokanson, 2023). Jika kebutuhan sistem salah didefinisikan sejak awal, maka arsitektur desain yang dibangun akan salah, kode program yang ditulis akan sia-sia, dan perangkat lunak yang dirilis akan ditolak oleh pengguna.

Rekayasa Kebutuhan (Requirements Engineering / RE) adalah proses terstruktur dan disiplin dalam menemukan, menganalisis, mendokumentasikan, dan memvalidasi layanan serta batasan operasional yang harus disediakan oleh sistem perangkat lunak (Sommerville, 2016).

3.2 Empat Tahapan Siklus Rekayasa Kebutuhan

Proses rekayasa kebutuhan mencakup 4 tahapan utama yang berjalan secara iteratif:

1. Elisitasi Kebutuhan (Requirements Elicitation): Aktivitas proaktif menggali kebutuhan dari berbagai pemangku kepentingan (Stakeholders) melalui teknik: Wawancara Mendalam (In-Depth Interviews), Lokakarya JAD (Joint Application Design), Observasi Alur Kerja Lapangan, Kuesioner Pengguna, Telaah Dokumen Regulasi Bisnis, dan Perumusan Skenario User Stories.

2. Analisis & Negosiasi Kebutuhan (Requirements Analysis & Negotiation): Mengurai kebutuhan yang tumpang tindih, mengidentifikasi kontradiksi antar-stakeholder, menilai kelayakan teknis dan biaya (Feasibility Study), serta menyepakati batasan ruang lingkup proyek.
3. Spesifikasi Kebutuhan (Requirements Specification): Menuangkan seluruh kebutuhan yang telah disepakati ke dalam dokumen formal Software Requirements Specification (SRS) yang terstruktur, tidak ambigu (Unambiguous), dan dapat diuji (Verifiable).
4. Validasi & Manajemen Kebutuhan (Requirements Validation & Management): Memeriksa dokumen SRS melalui Formal Inspection / Peer Review untuk memastikan kelengkapan, konsistensi, dan ketertelusuran (Traceability) terhadap tujuan bisnis.

3.3 Klasifikasi Kebutuhan: Fungsional vs. Non-Fungsional (Model FURPS+)

Kebutuhan perangkat lunak dibagi menjadi dua kategori fundamental:

A. Kebutuhan Fungsional (Functional Requirements):

Menjelaskan layanan, fungsi spesifik, komputasi data, dan alur proses apa yang HARUS disediakan oleh sistem (What the system shall do) saat menerima input tertentu (misal: 'Sistem harus mampu menghitung total potongan pajak PPh 21 secara otomatis saat kalkulasi gaji bulanan diproses').

B. Kebutuhan Non-Fungsional (Non-Functional Requirements / NFR):

Menjelaskan batasan kualitas, kinerja, keamanan, dan keandalan di mana sistem harus beroperasi (How well the system shall perform).

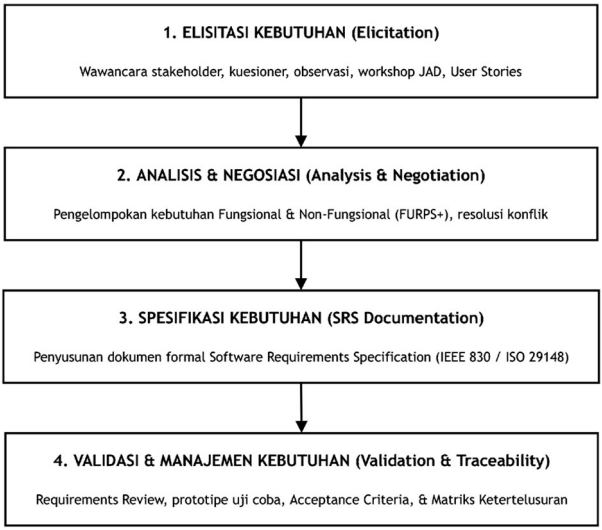
Model Klasifikasi Kualitas FURPS+ (Robert Grady / Hewlett-Packard):

- Functionality (Fungsionalitas): Kapabilitas fitur, keamanan autentikasi, dan kepatuhan aturan.
- Usability (Kebergunaan): Kemudahan pengoperasian antarmuka, ergonomi visual, dan dokumentasi bantuan.
- Reliability (Keandalan): Frekuensi kegagalan sistem (MTBF), akurasi output, dan ketersediaan (Availability 99.9%).
- Performance (Performa): Waktu respon transaksi (Response Time < 2 detik), throughput, dan konsumsi memori/CPU.
- Supportability (Dukungan Pemeliharaan): Kemudahan pemeliharaan kode (Maintainability), modularitas, dan portabilitas lintas platform.
- Simbol '+' (Plus): Batasan Desain (Design Constraints), Batasan Implementasi (Bahasa Pemrograman / Framework), Batasan Antarmuka (Integrasi API Eksternal), dan Batasan Fisik Server.

3.4 Struktur Dokumen Standar SRS (IEEE Std 830-1998)

Dokumen Software Requirements Specification (SRS) bertindak sebagai kontrak kerja resmi antara klien dan tim pengembang perangkat lunak. Format baku IEEE Std 830 memuat 3 bagian inti:

1. Bagian 1 - Pendahuluan (Introduction): Tujuan dokumen, Ruang lingkup produk sistem informasi, Definisi, Akronim, dan Referensi standar.
2. Bagian 2 - Deskripsi Umum Sistem (Overall Description): Perspektif produk terhadap sistem eksternal, Karakteristik fungsi utama produk, Karakteristik dan hak akses pengguna (User Classes), Asumsi ketergantungan sistem, dan Batasan umum desain.
3. Bagian 3 - Kebutuhan Spesifik (Specific Requirements): Deskripsi rinci kebutuhan fungsional (dilengkapi ID unik, Use Case, Input-Proses-Output), Kebutuhan antarmuka eksternal (UI, Hardware Interface, Software API Interface, Communication Protocols), dan Kebutuhan Non-Fungsional (Keamanan, Kinerja, Keandalan).



Gambar 3.1: Siklus Rekayasa Kebutuhan (Requirements Engineering - Sommerville & Wiegers)

Aktivitas Pembelajaran & Penugasan Terstruktur: LEMBAR KERJA 3 (TUGAS 3 RPS BAGIAN 1): Elisitasi Kebutuhan & Perumusan SRS

Lakukan elisitasi kebutuhan sistem informasi pada kasus nyata kelompok Anda, klasifikasikan kebutuhan Fungsional dan Non-Fungsional (FURPS+), dan susun draf dokumen SRS resmi (SubCPMK 3 | Bobot 20% terpadu).

Bagian A: Identifikasi Stakeholder & Elisitasi Kebutuhan Kasus

Komponen / Parameter Rekayasa		Deskripsi / Isian Analisis
1.	Judul Sistem Informasi yang Dikembangkan:	Sistem Informasi Manajemen Apotek & Distribusi Obat (SiApotek Enterprise)
2.	Profil Stakeholder Utama (Aktor):	Apoteker (Pengelola Resep), Kasir (Transaksi Pembayaran), Gudang (Stok Obat)
3.	Teknik Elisitasi yang Digunakan:	Wawancara Terstruktur kepada Kepala Apoteker & Observasi Alur Pelayanan Resep
4.	Kebutuhan Fungsional Kunci (SK-F-01):	Sistem wajib memberikan peringatan otomatis jika stok obat mendekati batas minimum
5.	Kebutuhan Fungsional Kunci (SK-F-02):	Sistem harus memvalidasi interaksi negatif antar-obat sebelum resep dicetak

Bagian B: Spesifikasi Kebutuhan Non-Fungsional (Model FURPS+)

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
Kebutuhan Usability (U):	Kasir pemula dapat menyelesaikan transaksi penjualan obat dalam waktu < 45 detik
Kebutuhan Reliability (R):	Sistem memiliki uptime ketersediaan minimum 99.8% selama jam operasional apotek
Kebutuhan Performance (P):	Waktu pencarian nama obat dari basis data 50.000 item tidak boleh > 500 milidetik
Kebutuhan Security (+):	Seluruh data transaksi dan riwayat resep pasien wajib dienkripsi AES-256
Status Kesiapan Draf Dokumen SRS:	Draf dokumen SRS terstruktur sesuai standar IEEE 830 siap dimodelkan ke UML

Lembar Refleksi & Evaluasi Proyek:

1. Jelaskan mengapa perubahan kebutuhan (Requirements Volatility) yang terjadi pada fase pengujian akhir (Testing) dapat menimbulkan biaya perbaikan yang berlipat ganda dibanding saat fase elisitasi:

.....

2. Bagaimana Matriks Ketertelusuran Kebutuhan (Requirements Traceability Matrix / RTM) membantu tim penguji memastikan bahwa seluruh butir kebutuhan dalam SRS telah diuji secara menyeluruh?

BAB 4: PERANCANGAN SISTEM & PEMODELAN PERANGKAT LUNAK BERBASIS UML

4.1 Prinsip-Prinsip Fundamental Perancangan Perangkat Lunak

Perancangan perangkat lunak (Software Design) adalah fase rekayasa di mana kebutuhan pengguna yang terdokumentasi dalam SRS ditransformasikan menjadi cetak biru teknis arsitektur sistem, struktur data, antarmuka komponen, dan algoritma prosedural sebelum penulisan kode dimulai (Pressman & Maxim, 2020).

Lima Prinsip Emas Desain Perangkat Lunak:

1. Abstraksi (Abstraction): Memfokuskan perhatian pada aspek-aspek esensial sistem tanpa terdistraksi oleh detail implementasi internal.
2. Modularitas (Modularity): Memecah sistem perangkat lunak yang besar menjadi unit-unit modul independen yang lebih kecil dan mudah dikelola.
3. Information Hiding (Penyembunyian Informasi - David Parnas): Setiap modul menyembunyikan detail logika internalnya dari modul lain dan hanya berkomunikasi melalui antarmuka publik yang telah didefinisikan secara ketat.
4. Kohesi Tinggi (High Cohesion): Seluruh elemen dan fungsi di dalam satu modul harus saling berkaitan erat dan bekerja sama untuk mencapai satu tujuan tunggal yang spesifik.
5. Kopling Longgar (Loose Coupling): Meminimalkan tingkat ketergantungan langsung antar-modul sehingga perubahan pada Modul A tidak merusak kinerja Modul B.

4.2 Taksonomi Unified Modeling Language (UML)

Unified Modeling Language (UML) dikembangkan oleh 'Three Amigos' (Grady Booch, James Rumbaugh, dan Ivar Jacobson) di bawah standarisasi Object Management Group (OMG) sebagai bahasa visual standar dunia untuk merancang dan mendokumentasikan sistem perangkat lunak berorientasi objek (Rosa & Shalahuddin, 2018).

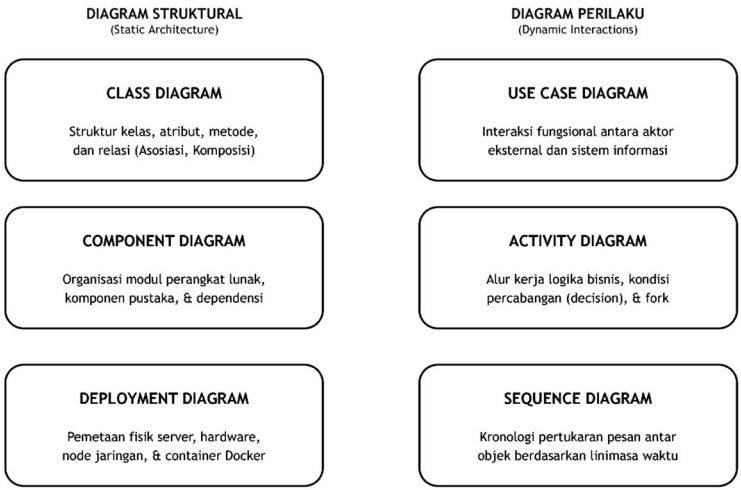
Diagram UML dikelompokkan ke dalam dua cabang besar:

A. Diagram Perilaku (Behavioral Diagrams): Menggambarkan aspek dinamis sistem, interaksi antar-objek, dan bagaimana sistem merespons rangsangan waktu.
B. Diagram Struktural (Structural Diagrams): Menggambarkan aspek statis sistem, komponen pembentuk, dan relasi logis/fisik antar-komponen.

4.3 Pemodelan Diagram Perilaku (Behavioral Diagrams)

1. Use Case Diagram: Memodelkan interaksi fungsional antara Aktor eksternal (pengguna/sistem lain) dengan batasan sistem (System Boundary):
    - Relasi `<<include>>`: Menandakan bahwa use case sumber PASTI dan SELALU memanggil use case target (misal: 'Proses Pembayaran' wajib `<<include>>` 'Validasi Saldo').
    - Relasi `<<extend>>`: Menandakan bahwa use case target dipanggil hanya pada kondisi tertentu / opsional (misal: 'Cetak Kuitansi Transaksi' adalah `<<extend>>` dari 'Proses Pembayaran').

2. Activity Diagram: Menggambarkan alur logika bisnis, urutan proses kerja, titik percabangan keputusan (Decision Nodes), dan proses paralel (Fork / Join) menggunakan jalur tanggung jawab (Swimlanes).
3. Sequence Diagram: Menggambarkan kronologi pertukaran pesan (Messages) antar-objek berdasarkan garis linimasa waktu vertikal (Lifelines) dan kotak aktivitas (Activation Bars).



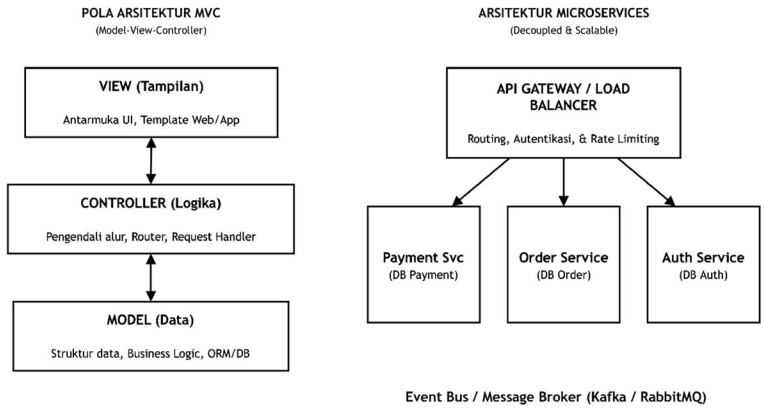
Gambar 4.1: Taksonomi Diagram UML (Diagram Struktural vs. Diagram Perilaku)

4.4 Pemodelan Diagram Struktural & Pola Arsitektur MVC

1. Class Diagram: Menggambarkan struktur statis kelas-kelas pembentuk sistem:
   - Tiga Bagian Kelas: Nama Kelas, Daftar Atribut (+ Public, - Private, # Protected), dan Daftar Operasi / Metode.

- Jenis Relasi Antar-Kelas: Asosiasi (keterhubungan biasa), Agregasi (relasi kepemilikan lemah 'has-a' di mana bagian dapat berdiri sendiri), Komposisi (relasi kepemilikan kuat 'part-of' di mana bagian akan musnah jika induknya dihapus), dan Generalisasi / Inheritance (pewarisan 'is-a').

2. Pola Arsitektur Perangkat Lunak:
   - Model-View-Controller (MVC): Memisahkan representasi data logika bisnis (Model), antarmuka tampilan grafis kepada pengguna (View), dan pengendali aliran logika pengatur permintaan (Controller).
   - Arsitektur Microservices: Memecah aplikasi monolitik raksasa menjadi layanan-layanan mikro mandiri yang saling berkomunikasi melalui protokol jaringan ringan (RESTful API / gRPC) via API Gateway.



Gambar 4.2: Pola Arsitektur Perangkat Lunak Modern (MVC Pattern vs. Microservices)

Aktivitas Pembelajaran & Penugasan Terstruktur: LEMBAR KERJA 4 (TUGAS 3 RPS BAGIAN 2): Pemodelan Sistem Berbasis UML

Rancang serangkaian diagram UML lengkap (Use Case, Activity, Class, Sequence) berdasarkan dokumen kebutuhan SRS yang telah disusun pada bab sebelumnya (SubCPMK 4 | Bobot 20% terpadu).

Bagian A: Identifikasi Komponen Use Case & Skenario Transaksi

Komponen / Parameter Rekayasa		Deskripsi / Isian Analisis
1.	Aktor Utama Sistem:	Apoteker, Kasir Pembayaran, Administrator Sistem, dan Pasien
2.	Daftar Use Case Inti:	UC-01: Autentikasi Login, UC-02: Kelola Stok Obat, UC-03: Transaksi Resep, UC-04: Laporan Keuangan
3.	Penerapan Relasi <<include>>:	Use Case 'Transaksi Resep' wajib <<include>> Use Case 'Cek Ketersediaan Stok'
4.	Penerapan Relasi <<extend>>:	Use Case 'Pemberian Diskon Member' meng-<<extend>> Use Case 'Transaksi Resep'
5.	Skenario Normal (Main Flow):	Kasir scan barcode obat -> Sistem tampilkan total harga -> Kasir terima pembayaran -> Struk dicetak

Bagian B: Spesifikasi Class Diagram & Desain Arsitektur

Komponen / Parameter Rekayasa	**Deskripsi / Isian Analisis**
Struktur Kelas Inti yang Dimodelkan:	Class Obat (Model), Class Transaksi (Model), Class KasirController, ViewFormKasir

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
Penerapan Encapsulation pada Atribut:	Atribut `kodeObat`, `namaObat`, `hargaJual` diset `- private` dengan getter/setter
Relasi Multiplicity Antar-Kelas:	Satu Objek Transaksi (1) memiliki banyak ItemTransaksi (1..*) -> Relasi Komposisi
Pola Arsitektur yang Dipilih:	Model-View-Controller (MVC) terintegrasi dengan REST API Backend
Status Kelengkapan Berkas Tugas 3:	Lengkap terlampir (Dokumen diagram UML terverifikasi untuk penilaian SubCPMK 4)

Lembar Refleksi & Evaluasi Proyek:

1. Jelaskan bagaimana pemodelan Sequence Diagram membantu programmer frontend dan backend memahami protokol pertukaran data API sebelum memulai pengkodean:

2. Mengapa prinsip 'High Cohesion dan Loose Coupling' sangat krusial dalam merancang arsitektur perangkat lunak berbasis Microservices?

BAB 5: MANAJEMEN PROYEK, PENGUJIAN PERANGKAT LUNAK, & PROYEK AKHIR SRS

5.1 Dimensi Manajemen Proyek Perangkat Lunak (Spektrum 4P)

Keberhasilan proyek perangkat lunak tidak hanya bergantung pada kecanggihan kakas pengkodean, melainkan sangat ditentukan oleh keandalan manajemen proyek (Software Project Management / SPM). Bob Hughes & Mike Cotterell (2009) mendefinisikan manajemen proyek perangkat lunak sebagai seni merencanakan, mengorganisasikan, memantau, dan mengendalikan sumber daya proyek agar sasaran kualitas, biaya, dan waktu dapat tercapai secara efektif.

Roger S. Pressman (2014) merumuskan Spektrum Manajemen 4P:

1. People (Manusia / SDM): Faktor paling krusial. Pengelolaan struktur tim pengembang, pembagian peran spesialis (Project Manager, System Analyst, UI/UX Designer, Programmer, QA Tester), kepemimpinan, dan motivasi kerja.
2. Product (Produk): Pemahaman ruang lingkup produk sistem informasi, batasan fungsionalitas, dan penetapan tujuan teknis.
3. Process (Proses): Pemilihan model SDLC yang sesuai (Waterfall, Scrum, dsb.) dan penentuan rangkaian aktivitas kerangka kerja.
4. Project (Proyek): Perencanaan pelacakan progres harian, manajemen risiko proyek, alokasi anggaran, dan kontrol kualitas.

5.2 Perencanaan, Estimasi Usaha, & Penjadwalan WBS

1. Work Breakdown Structure (WBS): Teknik dekomposisi hierarkis yang memecah seluruh ruang lingkup pekerjaan proyek menjadi paket-paket kerja yang lebih kecil dan terukur (Work Packages).
2. Estimasi Usaha & Biaya:
    - Function Point Analysis (FPA - Allan Albrecht): Menghitung ukuran perangkat lunak berdasarkan kompleksitas fungsi fungsionalitas eksternal (External Inputs, External Outputs, External Inquiries, Internal Logical Files, dan External Interface Files).
    - COCOMO II (Constructive Cost Model - Barry Boehm): Model matematis regresi untuk mengestimasi jumlah bulan-orang (Person-Months / PM) dan durasi waktu pengembangan.
3. Penjadwalan Proyek:
    - Gantt Chart: Visualisasi grafis batang horizontal yang memetakan jadwal mulai dan selesai setiap aktivitas proyek terhadap linimasa kalender.
    - Critical Path Method (CPM): Jalur terpanjang dari aktivitas-aktivitas yang tidak memiliki waktu luang (Zero Slack/Float); penundaan pada aktivitas di jalur kritis dipastikan akan menunda tanggal rilis seluruh proyek.

5.3 Strategi Pengujian Perangkat Lunak (Software Testing)

Pengujian perangkat lunak adalah proses mengeksekusi program dengan maksud menemukan cacat (Bugs / Errors) serta memverifikasi bahwa sistem memenuhi seluruh spesifikasi

kebutuhan dalam dokumen SRS (Pressman, 2014; Sommerville, 2016).

A. Tingkatan Hirarki Pengujian (Testing Pyramid):
   1. Unit Testing: Pengujian terhadap fungsi, metode, atau kelas individual secara terisolasi (misal: menggunakan framework JUnit pada bahasa Java atau PyTest pada Python).
   2. Integration Testing: Pengujian antarmuka dan komunikasi pertukaran data antarmodul saat digabungkan bersama.
   3. System Testing: Pengujian sistem perangkat lunak secara menyeluruh untuk memverifikasi fungsionalitas, performa beban, dan keamanan.
   4. User Acceptance Testing (UAT): Pengujian validasi akhir yang dilakukan oleh pengguna bisnis nyata berdasarkan skenario kerja aktual.

B. Metodologi Desain Kasus Uji (Test Case Design):
   - Black-Box Testing: Pengujian fungsionalitas eksternal tanpa melihat kode internal. Teknik utama: Equivalence Partitioning (membagi input ke dalam kelas valid dan invalid) serta Boundary Value Analysis (menguji nilai batas ekstrem minimum dan maksimum).
   - White-Box Testing: Pengujian struktur logika kode internal. Teknik utama: Basis Path Testing dan Cyclomatic Complexity ($V(G) = E - N + 2P$) untuk menjamin setiap jalur percabangan logika dieksekusi minimal satu kali.

5.4 Penyusunan Dokumen Portofolio SRS Proyek Akhir

Sebagai puncak evaluasi Mata Kuliah Rekayasa Perangkat Lunak (Bobot 40%), mahasiswa menyusun bundel dokumen portofolio profesional yang memuat:

1. Dokumen Software Requirements Specification (SRS) Berstandar IEEE Std 830.
2. Dokumentasi Diagram Pemodelan UML Lengkap (Use Case, Activity, Class, Sequence Diagram).
3. Lembar Rencana Pengujian Black-Box (Test Cases & Test Scenarios).
4. Perencanaan Manajemen Proyek (Struktur Tim, WBS, dan Linimasa Gantt Chart).
5. Presentasi Hasil Rancang Bangun Sistem di Hadapan Tim Dosen Penguji.

Aktivitas Pembelajaran & Penugasan Terstruktur: LEMBAR KERJA 5 (TUGAS 4 RPS & UAS): Rencana Pengujian & Portofolio SRS

Rancang tabel kasus uji fungsional Black-Box Testing, susun estimasi WBS proyek, dan lengkapi bundel dokumen SRS untuk evaluasi Ujian Akhir Semester (UAS - Bobot 40% kumulatif).

Bagian A: Matriks Kasus Uji Black-Box Testing (Boundary Value Analysis)

Komponen / Parameter Rekayasa		Deskripsi / Isian Analisis
1.	Fitur yang Diuji (Test Item):	Form Validasi Input Jumlah Pembelian Obat (Batas Valid: 1 s.d. 100 Butir)
2.	Kasus Uji Nilai Batas Bawah (Boundary Low):	Input: 0 (Ekspektasi: Muncul pesan error 'Jumlah minimal 1') -> Hasil: VALID
3.	Kasus Uji Nilai Batas Normal (Valid Normal):	Input: 50 (Ekspektasi: Transaksi berhasil diproses) -> Hasil: VALID
4.	Kasus Uji Nilai Batas Atas (Boundary High):	Input: 101 (Ekspektasi: Muncul pesan error 'Jumlah melebihi stok maks') -> Hasil: VALID
5.	Kesimpulan Hasil Eksekusi Uji:	Modul validasi input bebas bug dan memenuhi Acceptance Criteria SRS

Bagian B: Dekomposisi WBS & Checklist Portofolio Proyek Akhir UAS

Komponen / Parameter Rekayasa	Deskripsi / Isian Analisis
WBS Level 1 (Inisiasi & Analisis):	Wawancara stakeholder apotek, elisitasi kebutuhan, penyusunan draf SRS
WBS Level 2 (Perancangan Sistem):	Pemodelan Use Case, Activity, Class Diagram, dan Desain Basis Data
WBS Level 3 (Pengujian & Validasi):	Penyusunan Black-Box Test Cases, eksekusi UAT, dan perbaikan temuan
Kelengkapan Dokumen Portofolio SRS Final:	Lengkap & Terverifikasi sesuai standar IEEE 830 (Bab 1 - Bab 5)
Status Kesiapan Sidang Presentasi UAS:	100% Siap Dipresentasikan di Hadapan Tim Dosen Penguji

Lembar Refleksi & Evaluasi Proyek:

1. Jelaskan mengapa strategi pengujian bertingkat (Testing Pyramid) menempatkan Unit Testing sebagai lapisan pengujian paling banyak dibandingkan pengujian antarmuka (UI/E2E):

2. Susun dokumen portofolio komprehensif Rekayasa Perangkat Lunak kelompok Anda sebagai syarat mutlak kelulusan Ujian Akhir Semester (UAS)!

DAFTAR PUSTAKA UTAMA SESUAI RPS RESMI

Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach (8th ed.). New York: McGraw-Hill Education. [Pustaka Utama RPS]

Sommerville, I. (2016). Software Engineering (10th ed.). Boston, MA: Pearson Education Limited. [Pustaka Utama RPS]

Wiegers, K., & Hokanson, C. (2023). Software Requirements Essentials: Core Practices for Successful Business Analysis. Boston, MA: Addison-Wesley. [Pustaka Utama RPS]

Hughes, B., & Cotterell, M. (2009). Software Project Management (5th ed.). London: McGraw-Hill Higher Education. [Pustaka Utama RPS]

Rosa, A. S., & Shalahuddin, M. (2018). Rekayasa Perangkat Lunak: Terstruktur dan Berorientasi Objek. Bandung: Informatika. [Pustaka Utama RPS]

Sulistyowati, Andy R., & Nanang F.R. (2023). Rekayasa Perangkat Lunak: Memahami Proses, Metodologi, dan Tantangan. Surabaya: CV. Seribu Bintang. [Pustaka Utama RPS]

PROFIL DOSEN PENGEMBANG RPS & PENGAMPU

Sulistyowati, S.T., M.Kom.

Dosen Pengembang Rencana Pembelajaran Semester (RPS), Koordinator Rumpun Mata Kuliah Sistem Informasi, dan Dosen Pengampu Mata Kuliah Rekayasa Perangkat Lunak pada Program Studi S1 Sistem Informasi, Fakultas Teknik Elektro dan Teknologi Informasi, Institut Teknologi Adhi Tama Surabaya (ITATS)..

Rachman Arief, S.Kom., M.Kom.

Dosen Pengampu Mata Kuliah Rekayasa Perangkat Lunak dan Arsitektur Sistem Enterprise pada Program Studi S1 Sistem Informasi, Fakultas Teknik Elektro dan Teknologi Informasi, Institut Teknologi Adhi Tama Surabaya (ITATS).

Buku Ajar & Modul
Praktikum Perkuliahan

REKAYASA PERANGKAT LUNAK

Buku ini dirancang sebagai panduan komprehensif bagi mahasiswa dalam memahami teori dan praktik rekayasa perangkat lunak modern. Materi disusun secara sistematis mulai dari konsep dasar hingga penerapan model dan alat rekayasa perangkat lunak yang relevan dengan kebutuhan industri saat ini.

Ruang Lingkup Materi

- Prinsip Layered Technology
- Model SDLC: Waterfall, V-Model, Spiral, Agile Scrum
- Rekayasa Kebutuhan (SRS IEEE 830)
- Pemodelan UML
- Pengujian Perangkat Lunak (Black/White Box)
- Manajemen Proyek Berbasis RPS

[redacted]

Tentang Penulis

Penulis merupakan dosen dan praktisi di bidang Sistem Informasi yang memiliki pengalaman dalam pengajaran, penelitian, serta implementasi rekayasa perangkat lunak pada berbagai proyek berbasis teknologi informasi.

SISTEM INFORMASI

ITATS
INSTITUT TEKNOLOGI ADHI TAMA SURABAYA

Jl. Arif Rahman Hakim No. 100
Surabaya 60117 - Indonesia
www.itats.ac.id
info@itats.ac.id
(031) 594 5043

[redacted]